RADC-TR-89-58
Final Technical Report
May 1989



# ACCE NATURAL LANGUAGE INTERFACE INVESTIGATION

Odyssey Research Associates, Inc.

Dr. Richard I. Kittredge



ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-89-58 has been reviewed and is approved for publication.

APPROVED: Michael L. Mc Hale

MICHAEL L. Mc HALE
Project Engineer

APPROVED: Raymond P. Urtz

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER: James W. Hyde III

JAMES W. HYDE, III
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COES ) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS<br>N/A |
| 2a. SECURITY CLASSIFICATION AUTHORITY<br>N/A | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution unlimited. |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>N/A | 5. MONITORING ORGANIZATION REPORT NUMBER(S)<br>RADC-TR-89-58 |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Odyssey Research Associates, Inc. — 6b. OFFICE SYMBOL (If applicable) | 7a. NAME OF MONITORING ORGANIZATION<br>Rome Air Development Center |
| 6c. ADDRESS (City, State, and ZIP Code)<br>301A Harris B. Dates Drive<br>Ithaca, NY 14850-1313 | 7b. ADDRESS (City, State, and ZIP Code)<br>Griffiss AFB NY 13441-5700 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>Rome Air Development Center — 8b. OFFICE SYMBOL (If applicable)<br>COES | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>F41608-86-D-0010 |
| 8c. ADDRESS (City, State, and ZIP Code)<br>Griffiss AFB NY 13441-5700 | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO.: 62702F — PROJECT NO.: 5581 — TASK NO.: 27 — WORK UNIT ACCESSION NO.: 36 |
| 11. TITLE (Include Security Classification)<br>ACCE NATURAL LANGUAGE INTERFACE INVESTIGATION | |
| 12. PERSONAL AUTHOR(S)<br>Dr. Richard I. Kittredge | |
| 13a. TYPE OF REPORT<br>Final — 13b. TIME COVERED<br>FROM May 87 TO Dec 87 | 14. DATE OF REPORT (Year, Month, Day)<br>May 1989 — 15. PAGE COUNT<br>44 |
| 16. SUPPLEMENTARY NOTATION<br>Subcontract issued to ORA by Support Systems Associates, Inc. | |

| 17. COSATI CODES | | |
|---|---|---|
| FIELD | GROUP | SUB-GROUP |
| 12 | 05 | |
| | | |

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
ACCE
Prolog
Logic Programming
Natural Language Processing

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report provides a brief summary assessment of the strengths and weaknesses of logic programming languages such as Prolog for supporting the requirements for natural language processing in the Advanced Command and Control Environment (ACCE). In order to deal as precisely as possible with ACCE requirements, this report emphasizes one of the best understood ACCE goals: natural language access to databases.

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>Michael L. Mc Hale | 22b. TELEPHONE (Include Area Code)<br>(315) 330-2973 — 22c. OFFICE SYMBOL<br>RADC (COES) |

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

# 1 Prolog's Capabilities for Supporting ACCE Goals

This section provides a brief summary assessment of the strengths and weaknesses of logic programming languages such as Prolog for supporting the requirements for natural language processing in the ACCE program.

The ACCE program is oriented towards providing a powerful, flexible workstation for battle management, to be deployed within ten years. One of the primary features of the workstation is an intelligent, adaptive interface. The interface will allow an operator to communicate with an integrated set of computer systems using a combination of text (typed input), speech, gestures and tactile (e.g., mouse, touch-screen) input. The interface must adapt to a variety of types of user, representing quite distinct levels of expertise and needs. For each type of user and situation the interface must not only understand natural spoken and written English input. It must also be able to produce English output that is readily understandable by *the user in the context.*

At present, the functions of the ACCE workstation are stated rather generally. The operating requirements and constraints on the use of natural language which may be faced in particular fielded systems can only be surmised (see section 1.1 below). Some of the currently foreseen functions of the workstation include database query, expert analysis and planning systems, battlefield simulations and projections, and even language translation.



In order to deal as precisely as possible with ACCE requirements, we have carried out this investigation with an emphasis on one of the best-understood ACCE needs: natural language queries to databases. The proper understanding of queries in a natural dialog sequence is one of the most active areas of current research and development, with a variety of techniques and implemented systems already available. More important, it remains one of the most difficult language processing problems. Significant breakthroughs are needed in several problem areas if the language understanding goals of ACCE are to be fully met.

In the course of the last five years, the Prolog programming language has become one of the primary tools for building natural language processing systems. Prolog offers a number of interesting advantages, particularly during the design and prototyping phases of system development (see section 1.2.3 below). Moreover, as we will see below, the logic programming paradigm is now used for a large percentage of new approaches to natural language processing (NLP), both in the United States and abroad. It would therefore be impossible to gain a complete or balanced picture of the state of the art, or the prospects for fulfilling ACCE requirements, without giving full consideration to the fast-evolving tradition that Prolog represents.

## 1.1 Natural Language Requirements for the ACCE Intelligent Interface

The ACCE program is at an early stage where technologies are being evaluated. We must therefore be careful to distinguish several kinds of requirements for natural language interface applications:

- requirements relevant to testing and evaluation of technologies (e.g., clarity of code, portability of prototype systems between computers and adaptability to new problem areas);
- requirements concerning prototype design (e.g., ease and naturalness of mapping problems statements to prototype code);
- requirements relevant to system development, system maintenance and training of maintenance personnel (e.g., ease of reading code and learning coding techniques; productivity of programmers; modularity of programs);
- requirements relevant to performance of fielded systems (e.g., real-time constraints on software, limitations on hardware, etc.)

Clearly, these four areas have quite different requirements. Questions of code execution speed, for example, may be highly relevant to performance

in some real-time fielded systems, but hardly critical to testing, prototype design or training. On the other hand, programming languages which foster code perspicuity, clarity and modularity can greatly aid in system evaluation, prototype design, code maintenance and training. These features may be less relevant to performance of systems in the field.

In the near future the ACCE program must by necessity focus on the first two requirement areas. In the area of natural language, the emphasis must be on:

- evaluation of the current leading NLP technologies followed by acquisition and mastery by RADC personnel of systems incorporating those technologies;
- establishment of realistic specifications for the natural language functions of the ACCE interface on the basis of existing and emerging technologies;

### 1.1.1 Specific Natural Language Capabilities

To meet its basic goals, the ACCE natural language interface must have several specific capabilities including the following:

- the ability to handle vocabularies in excess of 1000 words, including expressions appropriate to various command and control sublanguages;
- broad coverage of English syntactic constructions used in command and control dialog exchanges;
- the ability to handle non-standard input such as idioms, elliptical expressions, sentence fragments, and grammatical errors;
- the robustness necessary to process correctly in the presence of recoverable errors (i.e., where redundancy of information in context allows reliable reconstitution of the intended input);

- the capability to fail "gracefully", with sufficient feedback to the user, when communication is not correctly received and not recoverable;
- the capacity to initiate and control "clarification dialogues" to elicit additional information from the user in certain cases of ambiguous or ill-formed input;
- the ability to control the focus of a conversation to maintain and change topic and subtopic as necessary;
- speech processing capabilities that allow short sentences to be input vocally, using a normal fluid rate of speech; there must be a close integration of speech processing abilities with the syntactic, semantic and pragmatic capacities of the interface (i.e., intelligent speech processing);
- the natural language interface should also be organized in such as way as to allow understanding and explanation using the full context of preceding input and output; in particular, query systems should be capable of interpreting elliptical queries or query fragments in the context of the preceding conversation; also any short output should be expandable or explainable, if additional information (such as knowledge of definitions) is required;
- understanding and generation of written language will frequently occur in the context of other media (such as graphics and speech, and even monitoring of and feedback to human sensory systems); special consideration must therefore be given to the problem of representing linguistic expressions in a way that can easily "communicate" with representations of expressions in other media.

## 1.2 Prolog's Relevance to ACCE

### 1.2.1 Assumptions

In this section we consider the way in which Prolog is particularly relevant to the requirements of the ACCE program and to some specific capabilities

envisioned for the ACCE natural language interface.

We assume here that ACCE interfaces, including their natural language components, must first be built in prototype form using one (or more) of the programming languages especially suited to symbolic processing such as Lisp or Prolog. It is likewise generally assumed that neither of these two symbolic processing languages is necessarily required for implementation of a fielded system.

The choice of implementation language for any given application must depend on trade-offs of speed, memory requirements, need to frequently modify or update the code, and other factors. We take the position here that by the time ACCE is ready for implementation the tradeoffs will have changed substantially and that the current need for evaluation and prototyping is all that can be safely considered.

We also take for granted that functional programming languages such as Lisp will continue to play an important role in NLP. Not only is there a substantial body of NLP experience accumulated during nearly 20 years of NLP programming in Lisp. There is also reason to believe that some NLP problems are more naturally treated as list processing in the functional paradigm, just as other NLP problems seem to be more naturally treatable in Prolog. Future programming languages for NLP may well combine strong points of both Lisp and Prolog (cf. LOGLISP and other extensions).

### 1.2.2 Two reasons for Prolog's importance

Our basic line of argument here is that Prolog is relevant for ACCE for two main reasons:

1. Prolog has features that are inherently suited to problems of natural language processing (NLP);
2. the rapid growth of Prolog as the language chosen to implement a wide variety of theories and accomplish a wide variety of tasks in

NLP means simply that a very substantial part of NLP technology is now accessible most directly within the logic programming paradigm.

Section 1.2.3 below summarizes some of the inherent features of Prolog that relate to NLP. Section 1.2.4 gives an overview of the depth and breadth of natural language processing applications now being carried out in Prolog.

### 1.2.3 Inherent Features of Prolog

Much of Prolog's success in NLP has come since the demonstration, by Pereira and Warren (1980) that parsing of syntactic structure can be viewed as a problem of logical deduction. Their introduction of definite clause grammars (DCGs) marked the real beginning of widespread usage of Prolog in NLP. [Note, however, that some fundamental features of DCGs were already present in Colmerauer's Q systems (1971), used as early as 1969 in the machine translation work of the TAUM group at Montreal University; Q-systems provided the basis for Colmerauer's Metamorphosis Grammars (1978), the immediate predecessor of DCGs]

DCGs provide a very natural, simple and powerful way of carrying out top-down, depth-first parsing on sentences. A DCG parser has no need to deal explicitly with "backtracking" during the essentially non-deterministic process of syntactic analysis. Problems of string manipulation are also handled in a way that is "hidden" from the grammar writer. This means that much of the complexity of programming an analyzer can be handled automatically by the DCG (and Prolog) interpreter. The programmer can concentrate on providing linguistic knowledge about grammatical constituents and properties of words in a natural declarative form.

It should be pointed out that DCGs do not require interpretation into Prolog programs, but only that such interpretation is both simple and natural.

Another important feature of Prolog for NLP is the ease with which logical structures are built up and manipulated within this language. Predicate logic has served as an important language for semantic representation in

a number of linguistic theories. Even semantic networks (which are two-dimensional and therefore not representable directly within machines) have a more primitive logical form. Logical structures are particularly useful for proving correctness of answers to database queries. They also provide a very direct way in which to express knowledge bases and inference rules required to carry out reasoning. Logic thus provides the essential glue linking linguistic and non-linguistic knowledge. Since Prolog is based so directly on logic, it is no accident that expert systems and database query systems are often much simpler when written in Prolog than in any other language. It is even possible to consider a Prolog program as being a set of inference rules and the Prolog interpreter as a particular inference engine, mirroring the structure of an expert system.

A final important feature of Prolog which is useful within and beyond natural language programming is the way in which it helps encourage a top-down decomposition of problems into subproblems.

### 1.2.4 Importance of Prolog Expertise within the Natural Language Processing Community

In this section we consider some of the dimensions of experience acquired to date in Prolog programming for NLP. The breadth and depth of this experience built up in only a few years can be taken as an indirect indication of the power of the logic programming paradigm. More directly, it simply means that Prolog has now become an important "lingua franca" for expressing and exchanging expertise about NLP.

Prolog has been applied to as wide a variety of NLP problems as has any other language. Its applications for DB query are too numerous to list here, but include substantial projects at several major US manufacturers of computer equipment. Recent applications to language generation include generation of English and French marine weather forecasts (Polguère, Bourbeau and Kittredge, 1987). Machine translation work at IBM-Yorktown Heights (McCord, 1986) and at New Mexico State University (Huang, 1985) has used Prolog, and the IBM system is now fairly large. Marseille Prolog has

been used for speech analysis and VM/Prolog for morphological analysis (Russo, 1987).

Prolog is routinely used for building and manipulating deep and surface syntactic trees, semantic nets, and other conceptual structures.

A wide variety of linguistic models have now been implemented in Prolog. The Second International Workshop on Natural Language Understanding and Logic Programming (Vancouver) presented implementations of Government-Binding theory (Stabler, 1987), String-Restriction Grammar (Dowding and Hirschman, 1987), and Discourse Representation Theory. Other theories being implemented in Prolog include Generalized Phrase Structure Grammar, Lexical Functional Grammar (Reyle and Frey, 1983; Koch, 1987), and Meaning-Text Theory (Iordanskaya and Polguère, 1987).

The size of natural language systems now being implemented in Prolog rivals that of systems in any other progrmming languages. The Darpa-supported Pundit system at Unisys, using Quintus Prolog, and the work at Yorktown Heights by McCord (1986), using VM/Prolog are just two examples. Several large projects are now underway in France and elsewhere in Europe within the Esprit program. In Japan, where Prolog was chosen for the machine language of the 5th Generation Computer project, dedicated hardware is being prepared which should allow systems to reach a new level of complexity and performance.

At least eight commercial Prologs have been used to write NLP systems on large or mid-size computers:

```
BIM Prolog (Belgian)
C-Prolog
DEC Prolog-10,20
Hewlett-Packard Prolog
M-Prolog
Prolog II (French)
Quintus Prolog
VM/Prolog
```

These are available on a wide variety of machines from micros to mainframes. Most major US computer manufacturers either market Prolog products (IBM,DEC,Symbolics[Prolog board],Hewlett-Packard) or else market machines for which a commercial Prolog is available (Sun, CDC). In addition, a number of foreign companies (e.g., Belgian BIM) either market their own Prologs or have them in the works.

In addition to the Japanese dedication to Prolog hardware in the 5th Generation Computing project, Japanese researchers have been active Prolog users and have recently begun to extend Prolog in new directions. For example, work by Ueda (1987) on guarded Horn clauses is aimed at adapting Prolog for parallel computer architectures.

Because Prolog originated in France and Britain, there is considerable Prolog expertise in Europe. French industry in particular is quite advanced in the use of Prolog for expert systems and NL interfaces (CGE-Marcoussis, Dassault, IBM-Paris). Three Soviet bloc countries, Hungary, Czechoslovakia and Poland, have been quite active in developing and using Prolog. Some early innovations in making modular programs occurred in Hungary, leading to the M-Prolog language which is marketed by Logicware in North America.

## 1.3 Some Current Limitations of Prolog

Prolog has not, in the short tradition up till now, been oriented towards efficient processing in real-time situations. Initial problems of control in general and constraining search in particular are only now being addressed on a broad scale by the logic programming community. Moreover, it is only recently that the design of hardware has been substantially influenced by the needs of logic programming, particularly in the context of the Japanese 5th Generation Project.

A number of efforts are underway to extend or modify Prolog to improve

the control problem. Several Prologs now allow modular communicating programs which limit rule application. A more substantial extension is the work on Meta-Prolog by Bowen (1985) at Syracuse University which will allow more flexible proof of propositions with respect to "theories" and "subtheories". Part of the control problem recedes with the advent of parallel architectures and the invention of new Prolog programming strategies to deal with them (e.g., guarded Horn clauses). Now that Prolog is firmly entrenched in North America, US hardware manufacturers (Motorola, TI) are seriously involved in the race to "put Prolog on a chip".

## 1.4 Conclusions

The preceding sections have summarized some of the strengths and weaknesses of logic programming languages such as Prolog for natural language programming of the kind envisioned in the ACCE intelligent interface. We have shown that much of the current work in NLP is based on Prolog. This is undoubtedly due in part to the ease with which (essentially declarative) knowledge of language can be expressed in logical form. It is also due to the way in which such problems as backtracking during analysis can be "hidden" from the system developer, thus cutting down on the complexity of the programming task. A further advantage is in the way in which Prolog encourages a top-down decomposition of the problems involved in language analysis.

It must be conceded that the experience with Prolog as a programming language for NLP is rather new, particularly in the U.S., where the tradition is scarcely five years old (compared to the Lisp tradition in NLP which is roughly 20 years old). Very large NLP systems are only now being written in Prolog, because "industrial-strength" implementations, with powerful development environments, have become available only in the past year. We should therefore expect to see a quantum jump in the average size of systems. Within two or three years, the appearance of new machine architectures more favorable to logic programming should also lead to a marked improvement in system performance.

One final point must be stressed. The success of the natural language functions in the intelligent ACCE interface does not depend crucially on the choice of language for prototyping and system building. It depends much more on the state of the underlying technologies, whatever the language they happen to be implemented in. It so happens that more and more interesting technologies are being implemented in Prolog.

# 2 Study of Chat-80 Technology

## 2.1 Introduction

This section deals with a study of the Chat-80 database query interface from the point of view of its technology for natural language processing (NLP). Particular attention was paid to the appropriateness of this technology for meeting the specific NLP requirements for language understanding in the ACCE program (see section 1).

Chat-80 represents a doctoral thesis work by Fernando Pereira (1983), and was originally written in DEC Prolog-10. It has been made available as a public domain program for several versions of Prolog which share the now standard Edinburgh syntax. The study of Chat reported in this section relied mostly on the Quintus Prolog version running on a VAX at RADC under VMS and on the (essentially identical) Quintus version running on Odyssey's Sun-3 under UNIX. There were apparently slight differences between the grammars of Chat-80 in the version discussed by Pereira and in the public release version (see discussion in 2.2.1 below).

Chat-80 was one of the first non-trivial NLP systems ever programmed in Prolog. At the time of publication the Chat analyzer was one of the largest to make use of the Definite Clause Grammar (DCG) formalism (Pereira and Warren, 1980), which simplifies the writing of computer grammars in Prolog. DCG rules are interpreted into pure Prolog for execution with other parts of the system written in Prolog. This made practicable for the first time a unified treatment of query analysis, query interpretation, database representation and query response, all through the use of programming in logic.

Chat-80 also introduced the so-called Extraposition Grammar formalism (see below). This allowed additional convenience in the grammatical bookkeeping required in processing certain semantically non-compositional constructions such as relative clauses.

The availability of Chat-80 as a part of the program examples package for

commercial Prologs has helped confirm Chat as an important milestone in NLP programming. Section 2.2 below gives examples of Chat's coverage of an important subset of English query constructions (interrogatives and, in principle, imperatives). In addition to being able to answer complex queries within this subset, Chat is also clear and elegant in its construction and modularity, and therefore relatively easy to adapt to similar domains (see section 2.6 below).

Despite its advantages, Chat remains primarily an initial reference point in the attempt to develop truly robust and powerful NL interfaces. Notwithstanding the importance of its core grammar, its grammatical coverage of English is still quite limited. Moreover, its inability to link the sentences of a dialog, as well as its separation of syntactic and semantic processing, would hinder its usage in any similar form as a component of the ACCE interface (section 2.4 below). These limitations can hardly be taken as criticisms, since Chat set a new standard in 1983, and it is only now that query systems are beginning to show substantially new capabilities. Instead, the technology contained in Chat should be viewed as an important stepping stone towards ACCE program goals.

## 2.2 Language Processing Capabilities and Limitations of Chat-80

Chat-80 is oriented towards answering queries about a geographical database of countries, cities, rivers, oceans and continents, and their properties and inter-relations. All of the countries and continents of the world are represented, as well as a selection of the more important capital cities and rivers.

Following are some sample queries (with answers) which Chat can parse, interpret and answer correctly. [Upper case letters are used here for readability]

```
1. "What is the largest country in Asia?"
    ("the Soviet Union")
```

```
2. "What is the maximum area of a country in Asia?"
    ("8347 ksqmiles")

3. "Do all continents contain countries?"
    ("no")

4. "Which continent does not contain a country?"
    ("Antarctica")

5. "What sea does the Rhone drain into?"
    ("the Mediterranean")

6. "What is the total area of countries in Asia?"
    ("18924 ksqmiles")

7. "What is the capital of Zaire?"
    ("Kinshasa")

8. "What countries does the continent which contains Zaire contain?"
    ("Algeria",....,"Zimbabwe")

9. "What are the continents no country in which contains more than
    two cities whose population exceeds 1 million?"
    ("[Africa,Antarctica,Australia]")

10. "Does any country contain two rivers?"
      ("yes")
```

Following are some queries which are not correctly processed by (the public release version of) Chat-80, but which one might expect to be processed given the domain. The reasons for failure to process correctly are quite various and are indicated below:

11. "What is the largest country in Asia and what is its capital?"
("I don't understand", i.e., no conjoined questions are parsed)

12. "What are the countries whose rivers drain
into the Mediterranean?"
("next question" : semantic failure probably due to ambiguity
of implicit quantifier on "rivers"-- all or at least one?)

13. "Is France or Germany in Asia?"
(failure to parse or interpret due to ambiguity)

14. "Is France in Europe or Asia?"
(failure to parse or interpret due to ambiguity)

15. "Which is larger, France or Germany?" (failure to parse)

16. "What is the total area of Asia?"
("nothing satisfies your question":
failure to correctly interpret due to usage of "total''
in this context, despite correct reply to (6) above)

17. "What are all the cities in Asia?"
(strange incorrect reply due to usage of "all the'')

18. "What is the distance from Rome to Paris?"
(failure to parse: data on distances not in database,
so syntax and semantics of verbs with two prepositions
"from ... to ...'' not covered in Chat)

19. "the largest country in Africa?"
(failure to parse, because questions must be
complete interrogative sentences)

### 2.2.1 Chat's strong points

On the positive side, it can be said that Chat's coverage of English is sufficient so that most simple well-formed questions which could be asked of the database can indeed be answered in at least one possible paraphrase form. The ability to correctly answer questions such as (9), which require substantial grammatical and semantic manipulation, was in fact quite impressive at the time of Chat's publication. Even five years after Chat, some laboratory prototypes of NL query systems still have difficulties on some of the queries (11)-(19).

According to Pereira (1983), Chat-80 can also handle a number of imperative and declarative sentence types. In the context of the original database, one would like to have imperatives such as:

```
"List the rivers in Asia!"  or
"Show me all the cities in Europe which are in countries
   through which the Danube flows."
```

Unfortunately, the public release version of Chat could not successfully parse any of the imperatives or declaratives indicated in the thesis version. It appears that some modifications were made before the public release occurred, and that these rendered inactive that part of the grammar. In the Charisma system (section 2.6), treatment of some imperatives was restored through the addition of postprocessing rules.

### 2.2.2 Linguistic limitations of Chat

On the negative side, at least from the point of view of the ACCE program, it must be said that Chat has several serious kinds of limitations. First, its coverage of the English grammar is not "dense" in the sense that many of the paraphrases of a correctly processed question fail to be parsed or

interpreted correctly. Moreover, users receive little feedback as to what led to the failure: the choice of failure message gives only implicitly the level (syntactic parsing, semantic interpretation, etc.) on which the failure occurred. The coverage also fails to be dense lexically: Chat's vocabulary is quite limited, giving few lexical alternatives for formulating equivalent questions (i.e., making semantic paraphrases).

Second, the coverage of English is indeed limited in some general ways. One surprising but easily repaired deficiency is the inability to take noun phrases and other sentence fragments followed by question marks as queries (cf. sentence 19 above). In continuous dialog, users of such commercial systems as INTELLECT (which is based on older technology than Chat) are allowed to use certain types of sentence fragments. Chat also does not allow the use of pronouns to refer to entities named in previous queries or replies, an almost essential need for dialog systems:

```
"What is the smallest country in Europe?"  ("San Marino")
"What is its area?"
```

Some more subtle deficiencies of Chat can also be observed. For example, the reply to (10) is not particularly helpful. A better reply would be:

```
"Yes, France and India."
 (according to the incomplete database)
```

Likewise, the reply to (3) forces a second question (4) to identify the questionable continent. A more "collaborative" response than (3) would be "No, Antarctica doesn't."

## 2.3 Some Design Characteristics of Chat-80

Fernando Pereira (1983) goes into some detail, in chapter 6, about Chat's general design limitations. He shows how extraposition grammars encounter serious difficulties in providing a general treatment of conjunction. Although Chat allows limited kinds of *and*-conjunction, including between relative clauses, each grammatical category must have its own conjunction rules. There are no meta-grammatical rules which apply to conjunction between any two constitutents of the same category (see Dahl and McCord, 1983).

Pereira also mentions the fact that top-down parsing, such as Chat provides, is inherently poor at providing analysis of sentence fragements, and in error recovery. At the same time, it is usually more efficient than bottom-up parsing for analyzing well-formed sentences.

Two other areas already earmarked for improvement in 1983 were the use of dictionary information in semantic interpretation and scoping and the resolution of anaphoric reference in (for example) pronouns.

## 2.4 Chat's Capabilities with respect to Specific ACCE Goals

It may be instructive to compare Chat's capabilities, as it is now configured, with the NLP processing goals of the ACCE intelligent interface (see section 1). It is conceivable that a database query system substantially like Chat could accomplish useful work in the map domain (see section 2.6 below). But to meet most of the general processing goals within the context of language understanding, substantially new systems must be designed.

Chat's vocabulary is limited to a few dozen words except for some three hundred proper names of countries, cities and rivers included for reference to database objects. The only verbs found in Chat were *border, contain, do, drain, exceed, flow, rise, have* and *be*. There are no problems in principle with expanding the vocabulary to the 1000-word range. But the amount

of lexical information associated with each word needs to be expanded substantially if words are to be allowed in all reasonable (grammatical) combinations which are meaningful for the domain. Chat is not necessarily optimally organized to accommodate lexicons or grammars which are large and complex.

As noted above, Chat cannot link successive sentences in dialog. There is no facility for interpreting anaphoric pronouns. Likewise, it cannot properly parse or interpret sentence fragments. It cannot initiate or control dialog in the way needed for clarification of misconceptions. The grammar is too limited to permit much ambiguity, and there is no mechanism for resolving any ambiguity that might arise in an expanded grammar. Although there is no control of conversational focus, the syntactic and logical representations used in Chat are sufficiently rich to provide a useful foundation for adding this and other conversational abilities. However a large part of a truly conversational system should be its facilities for generating replies to queries. Chat clearly has no facilities for generating full sentences much less extended texts.

One of the major deficiencies of Chat (and of most other NLP systems of the same vintage) is the inability to handle ill-formed queries. Mis-spellings, grammatical errors or semantically imprecise queries invariably lead to error messages or (in some cases) to the system's ignoring the input entirely. There is no attempt to "make sense" of an ill-formed query. (But note the addition of spelling correction in the Charisma system, derived from Chat). Such lack of robustness is typical of many research systems, but not to be tolerated in supposedly "user-friendly" environments. INTELLECT and other commercial database interfaces do rather well on this score, even to the point of correctly answering queries which have been typed in with the words in backwards order. But this feat seems entirely due to the limited possibilities for interpreting queries in specific databases. This is not a feature which will successfully "scale upwards" with a larger grammar and more complex domain.

## 2.5 Successors to Chat-80

Since early 1983, when Chat was published, a number of successor systems have been written. Some of these incorporated extensions or new approaches to problems met by Chat. We mention some of these to indicate the directions which have been taken in research. However, these extensions were not always tested in large systems or even integrated into systems which had all the capabilities of Chat. There is now a significant opportunity to combine some of these extensions and improvements in a new unified and "streamlined" system.

### 2.5.1 ORBIS

Within several months of the publication of Pereira's Chat-80, the ORBIS system was programmed by Colmerauer and Kittredge (1983) to demonstrate three features not available in Chat. First, ORBIS allowed queries to a database of astronomical objects (planets and their satellites) to be posed either in English or in French. A single dialog control component and database was used, along with parallel English and French grammar modules. The dialog component applied both grammars to the first sentence by the user (which could be in either language). The language whose grammar succeeded in parsing the question was then used by the system for creating replies. The dialog control was made possible by using the "freeze" predicate in Marseille Prolog II, which delays evaluation under certain conditions.

The bilingual feature of ORBIS suggests a way of introducing multiple-domain processing in future ACCE interfaces. Specialized sublanguage grammars (of English) could be run in parallel, controlled from a single dialog component, to try to fit the user's query to the proper domain.

A second innovation of ORBIS was its kind of feedback to the user. Any attempt to type a query which used words or structures outside the system's lexicon and grammar would trigger a message such as:

```
''Your sentence, which begins with 'What is the ...',
   should continue with one of the words:
   'diameter', 'mass', 'distance'."
```

The dialog component applied the grammar to generate all the word possibilities known to the system at the point where the user's query went astray. Although this would prove computationally expensive for systems with large grammars, such an approach might provide user friendliness for some systems where the number of words or grammatical categories is limited.

A third feature of ORBIS was its integration of syntactic and semantic processing within the same rules. Chat-80 separated the rules for computing syntactic structure from the rules for semantic interpretation, allowing for separate feedback and diagnostics from each processing stage. The integrated approach used by ORBIS and other more recent systems has three advantages: (1) it reduces the number of rules and (2) enforces the consistency of semantic coverage and syntactic coverage within the system. Furthermore, and perhaps most important, (3) this approach saves fruitless syntactic processing in cases where the semantic component of the rules cannot make sense of the syntactic analysis (see also Porto and Fulgueiras, 1984).

### 2.5.2 Modifier Structure Grammars

As indicated by Pereira in 1983, one of the serious drawbacks of the extraposition grammar approach was its inability to allow general treatment of conjunction. The processing of conjoined phrases by computer grammars has always been a major stumbling block. As early as the LUNAR system of Woods et al. (1972), considerable effort was put into assigning the proper scope to *and*-conjunction. When the ATN analyzer encountered an occurrence of *and*, all normal processing was halted. Special routines (SYSCONJ) were applied (see Woods, 1973), working out on both sides

from the conjunction until matching structures were identified.

Other builders of large grammars, including Sager (1981) at NYU using the string parser now also in use at Unisys, met the same problems and have likewise been forced into extraordinary efforts. Some researchers have even claimed that the problem of correctly scoping conjunctions is equal to the sum of all other problems in sentence analysis. This may be an exaggeration as far as syntax of database queries is concerned; the problem is far more frequent and difficult in expository scientific text. Nevertheless, correct conjunction scoping has remained a major challenge for sentence analyzers used in most interfaces.

One of the first improvements in treating conjunctions was given in Dahl and McCord (1983). Within their Modifier Structure Grammars (MSGs) it became possible to handle coordination metagrammatically, writing rules which applied across the board to conjoined expressions, independent of the particular (identical) grammatical categories conjoined.

Another important improvement introduced with MSGs was a flexible way of writing the semantic effect of syntactic constituent combinations within the syntactic parsing rule. This feature was later developed by McCord in his Modular Logic Grammars (see 2.5.5 below).

### 2.5.3 Gapping Grammars

Dahl and Abramson (1984) developed a different approach to handling problems such as extraposition and conjunction. Their Gapping Grammars (GGs) allowed the manipulation of more than one displaced element in arbitrary order. Pereira's extraposition grammars had been subject to a limitation on order called the "bracketing constraint", which corresponded to the ordering constraint in Woods' use of a HOLD list for ATNs.

### 2.5.4 SPH and INTERIX at French CGE Corp.

Alain Polguère (1984) built an analyzer for French, based on extraposition grammars and using the slot grammar method of McCord (1982). This was one of the first systems to interleave semantic structure building (including compositional treatment of quantifiers) with syntactic processing. More recently this system has been used by Stephan Guez as the basic sentence analyzer for his INTERIX system, a UNIX consultant expert system completed in 1987 at the French Compagnie Générale de L'Électricité.

### 2.5.5 Modular Logic Grammars

Michael McCord (1985,1986) has introduced the Modular Logic Grammar (MLG) formalism in the context of an experimental machine translation system (called LMT) being developed in VM/Prolog at IBM's Watson Research Center. MLGs are syntactically similar to DCGs, but with distinctions between strong and weak non-terminals, to help separate grammatical categories with semantic import from those which are used as auxiliaries during treatment of non-compositional structures. There are also logical terminals, used to build up pieces of semantic representation. Compiled MLG rules may apply in single-pass mode, where calls to semantics are interleaved with application of syntactic rules, giving only semantic (logical) forms as the output. Or they may apply in two-pass mode to build first a syntactic structure which is passed to the semantic interpreter.

MLGs allow special treatment of coordination and bracketing (i.e., the use of parenthetical-type structures) through metarules in the rule compiler. MLGs also help to hide a certain amount of rule complexity from the grammar builder.

## 2.6 Porting Chat-80 to the Map Domain

One of the tasks involved in evaluating Chat has been the collaborative RADC effort to modify the Chat-80 system for a domain of map informa-

tion (see McHale and Huntley, 1987). Although this new domain involves geographical knowledge, this knowledge is more "local" in that elementary objects are airstrips, towns, roads, railroads, power lines, map quadrants, etc.

To allow reference to the new ojectsin this different domain, it was necessary to add new nouns and special noun phrase types to the lexicon. In addition, new verbs and adjectives were needed to allow for expressing appropriate queries about the new domain. There also had to be a totally new semantic type hierarchy for the purposes of computing semantic compatibility of verbs and adjectives with their noun phrase arguments. In order to avoid major restructuring of the program during evaluation, no changes were made to the syntactic and semantic rules themselves, although certain special rules were added (see McHale and Huntley).

## 2.7 Summary and Conclusions

An investigation of Chat-80 from the point of view of extendability has shown that the overall system structure is quite amenable to making adjustments at the level of domain classes and vocabulary. A first experiment in adding prepositional complements to verbs in Chat proved easy thanks to the readability and modularity of the Prolog code.

Experiments in porting Chat to a new domain were also successful. Chat's grammatical coverage of English proved to be good enough for many queries needed in the map domain. Simple substitution of vocabulary (using a new hierarchy of semantic types) was sufficient, along with new database entries, to get started. Special problems then revolved around syntactic peculiarities encountered in the new domain (e.g., "type-2 obstruction"), or the addition of user-friendly features (spelling checker, help file) which did not interact with the complex syntactic and semantic rules. The resulting Charisma system (McHale and Huntley, 1987) was generally successful at giving responses to queries transposed to the map domain. Chat's readability and modularity greatly aided all these adjustments.

More ambitious changes were also initiated in Charisma. Experiments were conducted in adding imperatives, single noun phrase queries, and possessives and in improving the treatment of queries of the type *What is* (proper noun)?. Although it is generally possible to avoid modifying the syntactic core of Chat by making "post hoc" additions that graft onto existing rule sets, one gets the impression that the two-stage syntactic/semantic processing is too complexly stated to invite revision in its present form. Most of the same power of Chat's treatment of quantifiers and other difficult semantic problems could now be preserved in a version which would integrate syntactic and semantic processing in the same rules. Moreover, much of the complexity of these rules could be "hidden" from the grammar writer more successfully using recent techniques developed by McCord and others. Such a "streamlined" reformulation of Chat technology would facilitate the scaling up to larger domains.

The more serious systematic limitations in Chat identified in section 2.2 above have confirmed the impression gained from reviewing the NLP literature that much more remains to be done to meet the long-range needs of NL interfaces in the ACCE program. For the most part, the missing features (such as control of focus, anaphoric reference resolution and other dialog problems) are the subject of on-going research in a number of university and industrial laboratories. Experimental dialog systems are being developed in Prolog, Lisp and other languages. Progress depends not so much on the choice of programming tools, but rather on improving the basic linguistic data and representations, and on schemes for manipulating and co-ordinating various kinds of knowledge. Some early results from this research can now be added fairly easily to Prolog interfaces dealing with specific domains, such as the map domain of Charisma. The cumulated experience from several such applications is needed to foster the evolution of general domain-independent approaches needed for interfaces which can easily handle multiple, constantly changing application domains.

# 3 Strategies for Achieving ACCE Goals through Logic Programming

This section is devoted to outlining some strategies for promoting natural language processing technology that is specifically related to the goals of the ACCE intelligent interface. We assume here the logic programming paradigm as represented by the programming language Prolog. We further assume, as in the earlier sections, an emphasis on the goals of natural language understanding as needed for database query systems. For a discussion of Prolog and its relevance for ACCE goals in NLP, particularly in natural language understanding for interfaces, see section 1.

The separate options discussed in the sections below take as their common starting point the Chat-80 system of Fernando Pereira (1983) and its successors, discussed in section 2. These systems and approaches represent a fertile pool of NLP technology that is already available in the logic programming paradigm. The evaluation and testing of this technology with respect to ACCE database query goals is facilitated by Prolog's ready-made facilities for expressing relational databases, top-down parsers, and logical mechanisms for constructing and evaluating semantic representations. Still it should be stressed that the interest of this technology goes beyond the logic programming paradigm.

## 3.1 Integrating Syntax and Semantics

In order to build larger and more robust query systems than Chat, it is necessary to ensure a better integration of syntactic and semantic processing. The Chat-80 grammar covers a rather small, but crucial part of English syntax. The main emphasis is on interrogative structures containing simple sentences (not conjoined). Within the simple sentences, however, noun phrase syntax may be rather complex. Quantifier expressions and relative clauses may be (recursively) embedded, giving some strings whose syntactic and semantic analyses are not easily inter-related, such as the noun phrase in example (9), section 2.2:

*the continents no country in which contains more than two cities whose population exceeds 1 million*

The meaning of such an expression cannot be derived in a simple way from its syntactic structure and the meaning of its parts. In other words, the semantics of such an expression is not directly compositional. Pereira's approach to semantics in Chat is to derive a semantic representation for a whole (query) sentence from a previously computed syntactic structure. This has the advantage of keeping syntax and semantics more compartmentalized, and makes each individual rule a bit simpler. Syntactic rules might be debugged more directly. It also allows syntactically well-formed sentences to be recognized even when their meaning is not computable. This can give some feedback to a user about the source of ill-formedness, even if it may tend to slow down the total throughput of a system.

The approach often taken in successor systems to Chat is to interleave syntax and semantics in the same rule. One advantage of this approach is that it makes clearer the relationship between semantics and syntax in the system. It helps guarantee that when rules are changed, both aspects of the rule are considered at the same time. As the size of a system grows, this can become an important design principle. But most important, the interleaving approach ensures that syntactic/semantic rules are as efficient as possible. Putting semantic tests and logical structure building actions on each syntactic rule cuts down on unnecessary syntactic computation by blocking further processing when semantic well-formedness is not satisfied.

It is noteworthy that McCord's recent work (1986) on Modular Logic Grammars for large analyzers uses interleaving rules. In fact his interleaved rules can run in two modes: (1) interleaved calls to semantics or (2) separated computations. In applications such as machine translation where deep syntactic structure is often sufficient for language transfer, the semantic computation can be delayed or suppressed. In applications such as database query, where semantic computation is essential for proving that the reply is indeed a valid response, the tandem incremental computation of syntax and semantics can flag semantic ill-formedness at the earliest possible moment.

The trade-offs between interleaved and separated semantics in large systems are not yet well understood, despite the general arguments outlined above. It is therefore important to consider rewriting Chat, or its derivative system Charisma, in a form which uses interleaved rules. This may require some modifications in the approach to quantifiers and other semantically complex constructions, but should make the resulting system easier to extend in scope and expand in size. In addition, this rewriting should help to understand the generality of the scoping rules and other non-trivial semantic rules which lie at the heart of the semantic computation problem.

## 3.2 Grammatical Extensions

The grammatical coverage of Chat will be easier to extend following the rewriting process mentioned above. New domains such as the map domain inevitably require additions to the grammar as well as new lexical entries. Some of the query structures needed in Charisma, but which could not be added in a general way (without reconsidering the design of other rules) were:

*How far is Hudson Falls from Utica?*
(queries using scalar adjectives)

*What is the distance from Utica to Albany?*
(queries on nouns with prepositional complements)

*How much larger is Albany than Utica?*
(queries on measurable scalar comparatives)

*Does any railroad cross the Schroon River?*
(special semantics of new verbs)

One of the significant problems faced in a realistic implementation of the map domain is the heuristic definition of certain vague concepts (in order to give them a "working" meaning. These heuristic definitions need to interact with the more general semantic functions needed for quantifiers, relative clauses and conjunctions (to cite just three hard problems). A certain

amount of experimentation is needed to determine the ways in which the semantics of specific terms can interact with general semantic rules. The choice of terms used in a given domain can even impose limits or choice metrics on the overall semantic approach used.

Some of the vague terms requiring precise definitions that arose in determining the scope of Charisma were (vague term in italics):

> Does Albany *lie between* Burlington and New York?
> (how close to a straight line between Burlington
> and New York must Albany be?)
>
> Does route 9 pass *near* to Albany?
> (fuzzy definition for *near*)
>
> How *close* is Vermont to Philadelphia?
> (measuring between points and areas)
>
> What are *some* of the towns in Oneida County?"
> (fuzzy quantifier *some* in this context:
> how many towns should be named?
> should they all be listed if there aren't many?)

In cases such as spatial relations there should be consistent heuristic definitions for entire classes of words. Other vague terms will be rather specific to one word or a small class. In general these semantic definitions will not be verifiable directly in the database but rather through logical formulas. The point is that these formulas must be designed to interact correctly with the standard logical expressions which will not change from one application to another.

## 3.3 Dialog Capabilities

As the examples and discussion of section 2 indicate, there are many problems, and even dimensions of problems, which must be addressed in building dialog systems of the type needed for the ACCE program. Dialog control

and interpretation requires having access to both the syntax and the semantics of previous queries during the analysis of new queries. This in itself is an argument in favor of recasting Chat so that closer coordination is obtained between syntactic and semantic processing.

Because the number of dialog problems is large and because the general solution to these problems is not necessarily at hand, even in research systems, it is important for ACCE planning to assess in detail the specific dialog functions that are required in concrete systems. Some of the questions to be asked are:

- what sort of linguistic differences will there be between dialogs using typed input and those with spoken input?
- how detailed will system responses need to be?
  is there a need for multiple-sentence replies?
- what sorts of sentence fragments could be used as input, and how can these be ambiguous in particular contexts?
- to what extent will queries be answered by graphic displays or other non-linguistic feedback?
- to what extent will pronouns and other anaphoric expressions be used in the dialogs? what will be the typical and maximal scope of these expressions in referring to previous queries or replies in the current conversation, and in referring to objects or relations displayed graphically?

A series of simulation exercises using realistic data can help to identify specific problems and their relative frequency and other measures of importance. This should lead to an identification of the priorities within dialog handling. Study of commercial systems which have engineered ways to handle certain cases of anaphora and sentence fragments can help to identify potential needs. But care is needed in extrapolating the capabilities of current commercial systems towards semantically more robust systems.

Commercial systems often rely heavily on the semantic and pragmatic restrictions imposed by specific database predicates, and not on generalizable linguistic representations. In particular, the interplay between language and graphics in ACCE will require much better communication between the two data structures than is now possible in commercial database query systems. The representation of linguistic and non-linguistic knowledge must be richer by far to allow, for example, a pronoun in a query to be interpreted as referring to a highlighted object in the graphical reply to a preceding query.

Among the possible problems which should be high on the list of priorities we expect: pronouns and other deictic expressions, sentence fragments used as queries, and the problem of handling ill-formedness in the presence of solutions to these problems. Despite the difficulty of discourse problems, it is advisable to survey this terrain as early as possible to visualise future constraints on the general interface problem. In particular, we would urge that some initial work on fragments and pronominal anaphora be carried out on Charisma or a similar system derived from Chat. It would also be advisable to look at dialog forms needed for future spoken input as well as the forms currently needed for typed input.

## 3.4 Integrated Interface Requirements: Speech and Graphics

A multi-media interface of the kind needed by the ACCE program puts a number of design constraints on each module. In particular, the representations used for linguistic meanings must communicate with those used for graphical entities. This constitutes a new requirement over and above a NL query system as conceived in Chat, which assumes a "stand-alone" system.

The Charisma extension to Chat has adjusted the domain towards one in which graphical output can serve as the response to queries. In the light of this experience, two further extensions become possible, (1) the addition spoken input, and (2) the co-ordination of spoken input with mouse input.

First, in a restricted domain such as the Charisma map domain or the original Chat geographic database domain it is possible to add spoken input as an option. For the most part, the vocabulary and grammar of queries is restricted enough to allow word recognition using existing commercial speech recognizer. However, two problems must be solved. First, existing recognizers based on individual word recognition (e.g., Votan VPC 2000) can make too many recognition errors, even within a vocabulary of 50-100 words to be useful directly. However, these may give satisfactory performance if their output can be filtered through grammatical and semantic contextual constraints which then make possible error recovery strategies. Recognizers allowing in-built finite state grammars (e.g., Verbex, ITT) might allow some grammatical constraints to be stated, but have no capability for stating the context-sensitive restrictions on a grammar of the power used by Chat. Error recovery procedures which add context-sensitivity to word-based recognition do exist (e.g., Dreizin,1986) and can be adapted to enhance finite-state grammar recognizers. But one additional problem occurs: current recognizers cannot cope with the large branching factor presented by proper names in queries. Particularly in the case of foreign place names, the word templates will be insufficiently reliable. And since usually there are hundreds of possible place names at particular points in map queries, this area of recognition appears difficult.

In order to successfully add speech input to a system like Charisma it would be possible to circumvent the place-name problem in the following way. In the context of map displays on which place names are already indicated, it would be possible to speak the query using deictic place adverbs (e.g., *here,there*) co-ordinated with clicking of a mouse or other input device on the portion of the map containing a name or other location label. Not only would this give more reliable input (no more need to pronounce and perhaps mispronounce foreign place names), but the mouse input would confirm certain aspects of the syntactic structure of the query.

## 3.5 Recommendations

Among the strategies proposed above, there seem to be some ordering constraints. The problem of co-ordinating syntactic and semantic processing through a single set of rules appears to take priority. In the process of recasting the syntax/semantics interaction, certain additional problems posed by dialog can be taken into consideration. Once a rebuilt system is running, additional grammatical structures can be added and specific dialog capabilities tested. In order to set up a program for incremental improvement of dialog processing, however, it is important to collect dialogs from simulated man/machine interactions with a realistic problem set.

In case specific domain simulations from command and control are not available in the near future, the map domain should be taken as an object of serious simulation study. Man-machine interactions in this domain appear to present many of the interesting natural language problems that should arise in command and control. Perhaps the additional element required, and which is absent in Chat, is some model of the speaker and his goals. Before adding this additional dimension to interface design, it would be advisable to have a solid basis in the processing of more "objective" queries. The approaches of Chat and its successor systems, adapted to domains such as Charisma's map domain, provide an excellent starting point from which to build.

# 4 REFERENCES


Bates,Madeleine and Weischedel,R. (1987) Evaluating Natural Language Interfaces (manual for tutorial at 25th Annual Meeting of the ACL, Stanford).

Bowen,Kenneth (1985) Meta-Level Programming and Knowledge Representation. Tech.Rep. CIS-85-1, School of Computer and Information Science, Syracuse University.

Bowen,Kenneth and Weinberg,T. (1985) A Meta-Level Extension of Prolog. in Proceedings of the 1985 Symposium on Logic Programming, eds.Cohen,J., and Conery,J., IEEE Computer Society Press, Washington D.C., pp. 48-53.

Colmerauer,Alain (1971) ''Les systèmes-Q: un formalisme pour analyser et synthetiser des phrases sur ordinateur'', Internal Report, TAUM Group, Universite de Montreal.

Colmerauer,Alain (1978) ''Metamorphosis Grammars'', in L.Bolc, ed., Lecture Notes in Computer Science, vol.63,pp.133-189, Springer Verlag.

Colmerauer,Alain and Kittredge,R. (1983) ORBIS (bilingual English-French database query system demonstrated at IJCAI-83 conference)

Dahl,Deborah (1986) ''Focussing and Reference Resolution in Pundit'', Proceedings of AAAI-86 (Fifth National Conference on Artificial Intelligence), pp.1083-1088.

Dahl,Veronica and Abramson,H. (1984) ''On Gapping Grammars'', Proceedings of the Second International Logic Programming Conference, Uppsala, pp.77-88.

Dahl,Veronica and McCord,M. (1983) ''Treating Co-Ordination in Logic Grammars'', American Journal of Computational Linguistics, vol.9, pp.69-91.

Dahl,Veronica and Saint-Dizier,P.,eds. (1985) Natural Language Understanding and Logic Programming (Proceedings of the First Int'l Workshop on Natural Language Understanding and Logic Programming, Rennes). North-Holland.

Dowding,John and Hirschman,L. (1987) ''A Dynamic Translator for Rule Pruning in Restriction Grammar'' in Saint-Dizier(ed.)

Dreizin,Felix, Kittredge,R. and Korelsky,T. (1986) ''Semantic Techniques for Error Recovery: An Application to Fire Control Dialogs'' Proc. Military Speech Tech Conference, Arlington.

Grosz,Barbara, Appelt,D., Martin,P. and Pereira,F. (1987) ''TEAM: An Experiment in the Design of Transportable Natural-Language Inter-faces'', Artificial Intelligence 32, pp.173-243.

Hirschman,Lynette (1987) ''Conjunction in Meta-Restriction Grammar'', Journal of Logic Programming.

Huang,X-M. (1987) ''Machine Translation in SDCG Formalism'', in Nirenburg,S.(ed.)

Iordanskaya,Lidiya and Polguère,A. (1987) Generation of Reports on the Activity of an Operating System Using Conceptual Communicative Representations. Technical memo, Odyssey Research Associates, Montreal.

Koch,Gregers (1987) ''Computational Logico-Semantic Induction'', in Saint Dizier,ed.

Matsumoto,Y. (1987) ''A Parallel Parsing System for Natural Language Analysis'', New Generation Computing, 5:63-78, Springer-Verlag.

Matsumoto,Y., Tanaka,H., and Kiyono,M. (1986) ''BUP: A Bottom-Up Parsing System for Natural Languages'', in van Caneghem,M and Warren,D.,eds.

McCord,Michael (1982) ''Using Slots and Modifiers in Logic Grammars for Natural Language'', Artificial Intelligence, vol.18, pp.327-367.

McCord,Michael (1985) ''Modular Logic Grammars'', Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics,Chicago.

McCord,Michael (1986) ''Design of a Prolog-Based Machine Translation System'', Proceedings of the Third International Logic Programming Conference, Springer-Verlag, Lecture Notes in Computer Science.

McHale,Michael and Huntley,M. (1987) ''Charisma'' Internal Report, RADC.

Nirenburg,Sergei,ed. (1987) Machine Translation: Theoretical and Methodological Issues. Cambridge University Press, Studies in Natural Language Processing.

Pereira,Fernando (1983) Logic for Natural Language Analysis. Technical Note 275, SRI International.

Pereira,F. and Warren,D.H.D. (1980) ''Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Transition Networks'', Artificial Intelligence, vol.13, pp.231-278.

Pereira,Fernando and Warren,D.H.D. (1983) ''Parsing as Deduction'', Proc. 21st Annual Meeting of the Assoc.for Computational Linguistics, pp.137-144.

Polguère,Alain (1984) Programmation logique des interfaces langue naturelle [Logic programming of NL interfaces]. Internal report, Marcoussis Laboratories, Research Centre of the General Electric Company [of France] (CRCGE).

Polguère,Alain, Bourbeau,L. and Kittredge,R. (1987) RAREAS-2: Bilingual Synthesis of English and French Marine Weather Forecasts. Technical memo, Odyssey Research Associates, Montreal.

Porto,Antonio and Filgueiras,M. (1984) ''Natural Language Semantics: A Logic Programming Approach'', Proceedings of the IEEE International Symposium on Logic Programming, Atlantic City.

Reyle,Uwe and Frey,W. (1983) ''A Prolog Implementation of Lexical Functional Grammar'', Proc. IJCAI-83, pp.693-695.

Russo,Marina (1987) ''A Rule-Based System for the Morphologic and Morpho-Syntactic Analysis of the Italian Language'' in Saint-Dizier(ed.)

Sager,Naomi (1981) Natural Language Information Processing, Addison-Wesley.

Saint-Dizier,P.,ed. (1987) Proceedings of the Second International Workshop on Natural Language Understanding and Logic Programming, Simon Fraser University.

Stabler,Edward (1987) ''Parsing with Explicit Representations of Syntactic Constraints''

in Saint-Dizier (ed.).

Ueda, Kazunori (1987) Guarded Horn Clauses. MIT Press

van Caneghem,M. and Warren,D.H.D. (1986) Logic Programming and Its Applications. Ablex.

Walker,A.,ed., McCord,M., Sowa,J. and Wilson,W. (1987) Knowledge Systems and Prolog. Addison-Wesley.

Woods,William, Kaplan,R. and Nash-Webber,B. (1972) The Lunar Sciences Natural Language Information System: Final Report. report 3438, Bolt, Beranek and Newman Inc.

Woods,William (1973) ''An Experimental Parsing System for Transition Network Grammars'', in R.Rustin,ed., Natural Language Processing. Algorithmics Press.